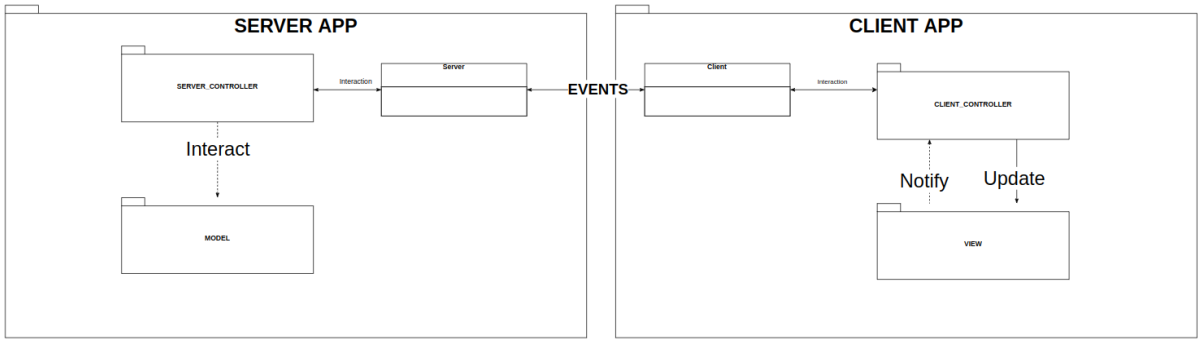
Descrizione UML Avanzato

Gruppo GC6 (Rigamonti-Rogora-Sarbu)

1. Introduzione

Abbiamo continuato ad implementare le varie parti dell' MVC, con l'aggiunta di un Event System che mira a sostituire il deprecato pattern "Observer".
Nei vari diagrammi in allegato è stata adottata una suddivisione in pacchetti onde evitare la confusione generata da un diagramma troppo esteso.



In seguito verranno descritti i vari pacchetti:

2. GameModel

Il modello è stato variato poco dall'ultima review, eccetto la gestione degli effetti dei personaggi, la quale non avviene più nel modello ma vengono gestiti in modo dinamico con delle Strategy nella GameLobby.
Sono stati aggiunti più getter per permettere di inviare i vari elementi del Model senza dover inviare il modello completo.
Abbiamo modificato anche il passaggio della modalità di gioco alla schoolboard, così da evitare di creare un' istanza di CoinSupply in Normal Mode.

3. Controller lato Server

Dato che abbiamo deciso sin da subito di implementare la funzionalità delle partite multiple, abbiamo deciso di adattare il Server Controller per poter contenere più "GameLobby" all'interno di una Map la cui chiave è un codice univoco autogenerato, che i player possono usare per connettersi alla stessa partita.
All'interno di ogni GameLobby sono istanziate più classi atte a gestire le varie fasi di gioco. Alcune di esse presentano una Strategy data dagli effetti passivi dei personaggi che modificano quella specifica fase.

4. Networking

4.1 Server

Il Server presenta al suo interno una mappa contenente come chiave il nome del giocatore e come valore un ClientHandler. Il ClientHandler è responsabile della

ricezione e dell’invio degli eventi sulla rete. Il ClientHandler quando riceve un evento lo pubblica su una queue sincronizzata presente sul server.
Il server si occupa degli eventi sequenzialmente in modalità FIFO.

4.2 Client

Il client presenta al suo interno una classe ServerConnection che si occupa dell’invio e della ricezione degli eventi sulla rete.

5. Event System

L’event system è caratterizzato da una classe singleton chiamata EventManager presente sia sul client che sul server. Una classe può registrare dei callback sull’event manager solamente se implementa l’interfaccia EventListener.
Quando la classe si registra sull’EventManager quest’ultimo filtra le funzioni di callback in base all'annotazione @EventHandler.
Per inviare un evento ai vari listener basta quindi eseguire EventManager.notify passando l’evento da notificare.

5.1 Lobby event

L’annotazione EventHandler accetta una stringa come parametro, il topic.
Questo serve all’EventManager presente sul server ad inoltrare alla GameLobby giusta un evento.

6. Controller lato Client

Nel controller lato client abbiamo aggiunto metodi che aggiornano il Light Model, al quale la view fa riferimento. Sono presenti inoltre metodi che invocano le varie funzioni della view al momento opportuno (Ad esempio quando un client riceve un evento che comunica L’inizio del turno di un altro giocatore, invoca un metodo della view, viewinstance.displayMessage(//messaggio), che lo stampa a video nella CLI e lo mostrerà a suo modo nella GUI).

7. View

La view Presenta una classe astratta, dove viene effettuato un @Override dei metodi dalla CliView e dalla GuiView, così da rispondere agli stessi eventi con diverse implementazioni in base alla modalità di visualizzazione adottata. Dopo le varie azioni avvenute nella view e generate dall’interazione con l’utente, essa effettua una “Notify” al client controller che le comunica tramite eventi al server.